Combining the Benefits of Learning and Optimization

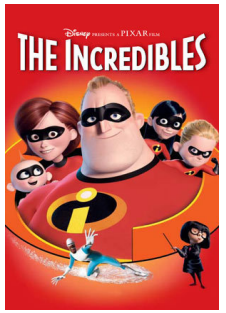
Aravind Rajeswaran

University of Washington

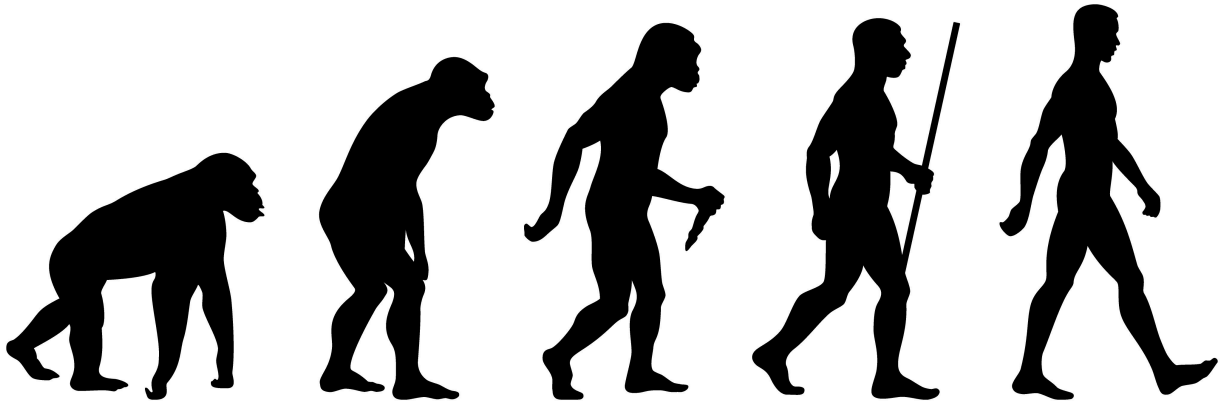
How to make embodied agents move like humans and acquire a vast repertoire of skills?

Many exciting application domains:

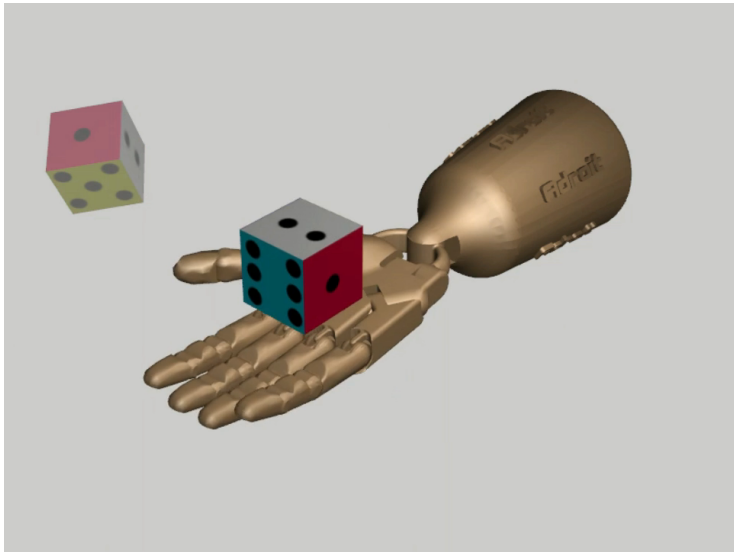
robotics, animation, biomechanics



Human behaviors are shaped by iterative improvement principles



Evolution, learning, adaptation



Key to automation in robotics is developing

efficient optimization algorithms for learning and control

Hard Optimal Control with Generative Models

Setting: Solving hard optimal control problems when we have a generative model (simulator)

(A motivating example)

OpenAI dactyl system

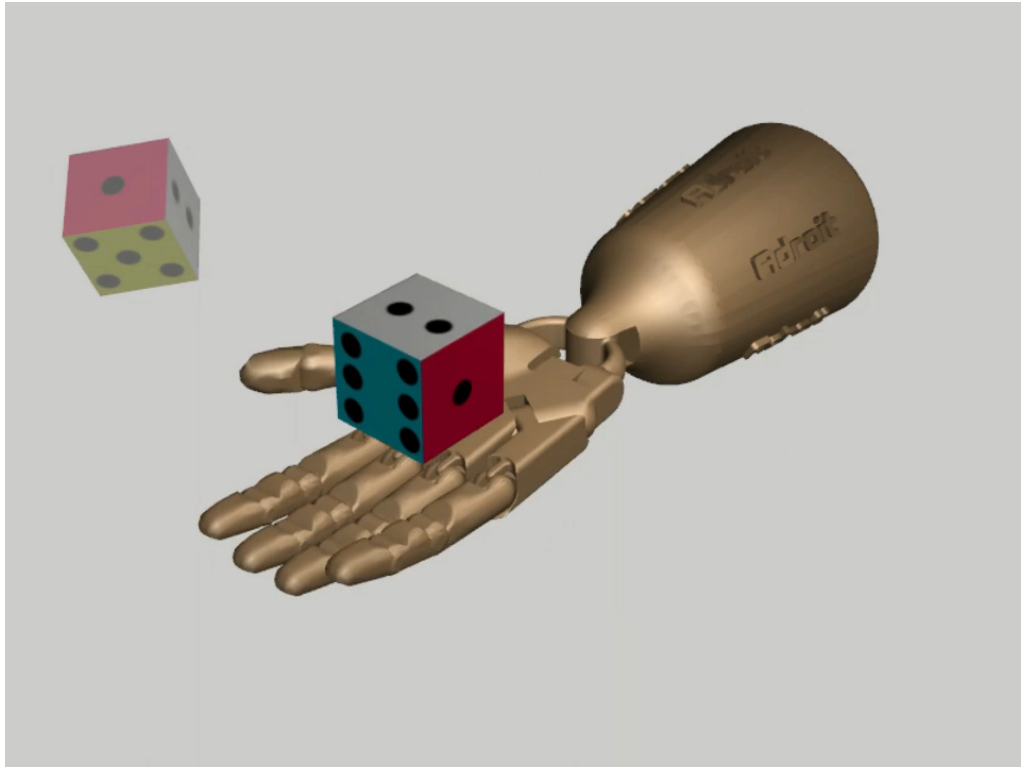
- Sim2Real with MuJoCo (and randomization)
- Policies trained with PPO
- "Rapid" compute framework (made for DOTA)
- 6144 CPU cores and 8 GPUs over a cluster
- 100 years of simulated experience
- 50 wall clock hours with 1000s of CPUs

Amazing feat of engineering and an extremely impressive robotics result!

Can we improve efficiency through algorithms that use the simulator more integrally?

Plan Online Learn Offline (POLO)

POLO framework allows for solving hard control problems like *dexterous hand manipulation* and *humanoid locomotion* in under *1 hour on a laptop!* (orders of magnitude more efficient than policy grad)



Make hand manipulate cube to match desired cube configuration (on the left)

POLO combines three main pillars

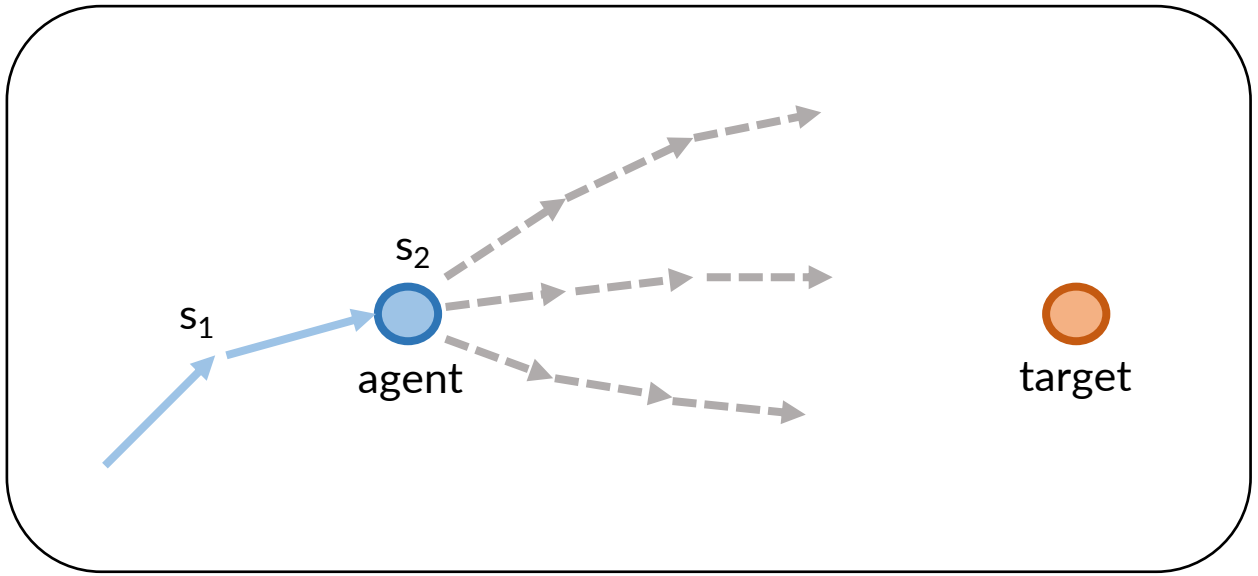
- *Online optimization* (model predictive control) for fast and efficient improvements
- *Consolidation of experience* (value function learning) to enable faster and longer horizon planning and reasoning
- *Directed exploration* (planning to explore) to efficiently discover optimal behaviors

Synergistic relationship between pillars allows their combination to overcome their individual limitations.

Model Predictive Control

Model Predictive Control (MPC) is extremely successful in many control applications including robotics

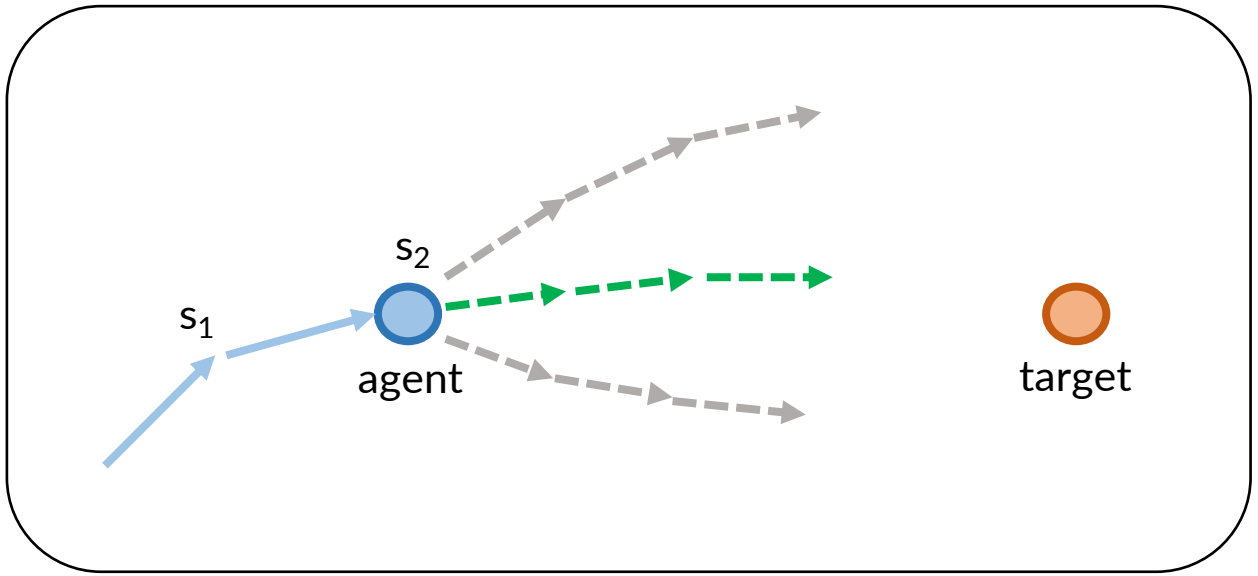
$$
\begin{aligned}
\underset{u_0,u_1,\cdots,u_H}{\text{maximize}} \quad & \mathbb{E}\left[\sum_{k=0}^{H}\gamma^k r(x_k,u_k)\right] && \text{(Trajectory Cost)} \\
\text{subject to} \quad & x_{k+1}\sim P(\cdot|x_k,u_k) && \text{(Dynamics)} \\
& x_0 = s_t && \text{(Initial State)} \\
\text{Result} \quad & a_t \sim \pi_{MPC}(\cdot|s_t) \equiv u_0 && \text{(Solution)}
\end{aligned}
$$



Model Predictive Control

Model Predictive Control (MPC) is extremely successful in many control applications including robotics

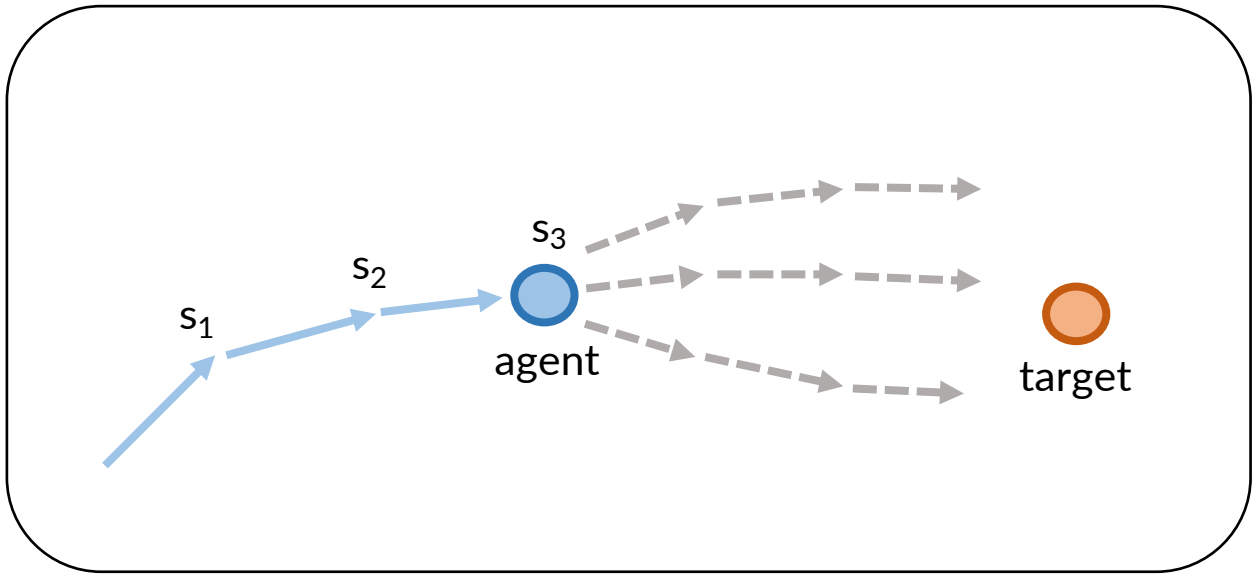
$$
\begin{aligned}
\underset{u_0,u_1,\cdots,u_H}{\text{maximize}} \quad & \mathbb{E}\left[\sum_{k=0}^{H} \gamma^k r(x_k, u_k)\right] && \text{(Trajectory Cost)} \\
\text{subject to} \quad & x_{k+1} \sim P(\cdot|x_k, u_k) && \text{(Dynamics)} \\
& x_0 = s_t && \text{(Initial State)} \\
\text{Result} \quad & a_t \sim \pi_{MPC}(\cdot|s_t) \equiv u_0 && \text{(Solution)}
\end{aligned}
$$



Model Predictive Control

Model Predictive Control (MPC) is extremely successful in many control applications including robotics

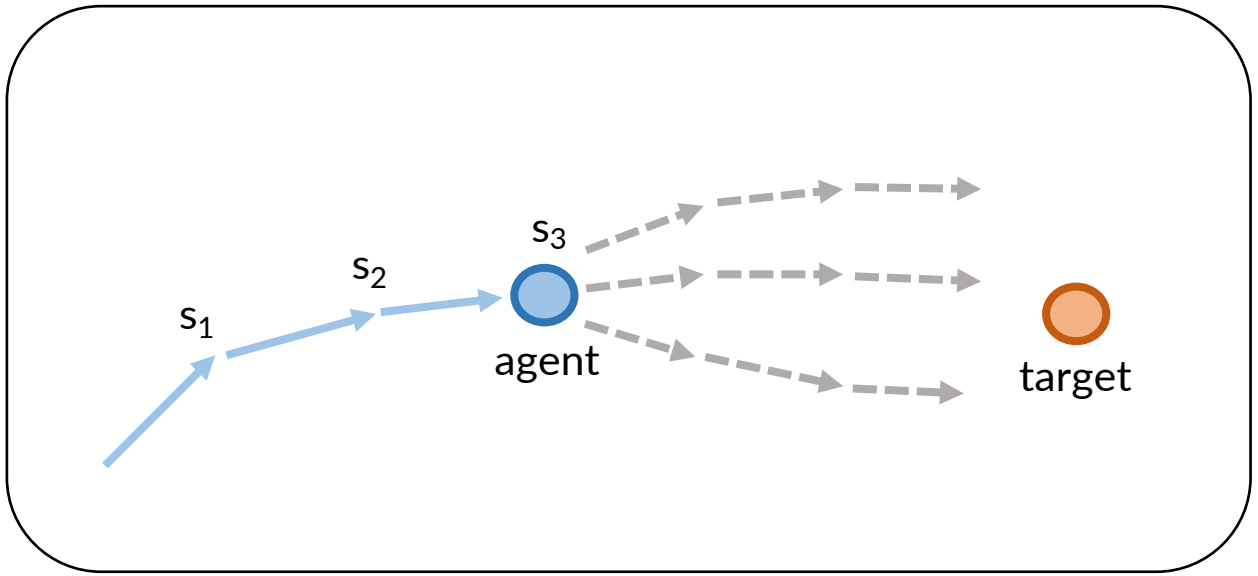
$$
\begin{aligned}
\underset{u_0,u_1,\cdots,u_H}{\text{maximize}} \quad & \mathbb{E}\left[\sum_{k=0}^{H}\gamma^k r(x_k,u_k)\right] && \text{(Trajectory Cost)} \\
\text{subject to} \quad & x_{k+1}\sim P(\cdot|x_k,u_k) && \text{(Dynamics)} \\
& x_0 = s_t && \text{(Initial State)} \\
\text{Result} \quad & a_t \sim \pi_{MPC}(\cdot|s_t) \equiv u_0 && \text{(Solution)}
\end{aligned}
$$



Model Predictive Control

Model Predictive Control (MPC) is extremely successful in many control applications including robotics

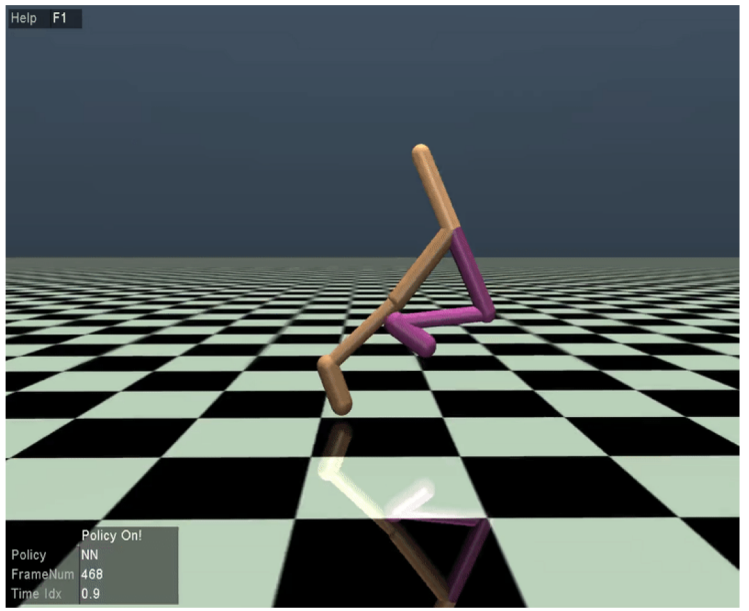
$$\begin{aligned}
&\underset{u_0,u_1,\cdots,u_H}{\text{maximize}} && \mathbb{E}\left[\sum_{k=0}^{H}\gamma^k r(x_k,u_k)\right] && \text{(Trajectory Cost)}\\
&\text{subject to} && x_{k+1}\sim P(\cdot|x_k,u_k) && \text{(Dynamics)}\\
& && x_0 = s_t && \text{(Initial State)}\\
&\text{Result} && a_t\sim\pi_{MPC}(\cdot|s_t)\equiv u_0 && \text{(Solution)}
\end{aligned}$$



Instance-specific, local, and fully online approach

Advantages:

- More expressive power than function approximation (non-parametric vs parametric)
- Distribution agnostic and works anywhere, while function approximation struggles out of distribution

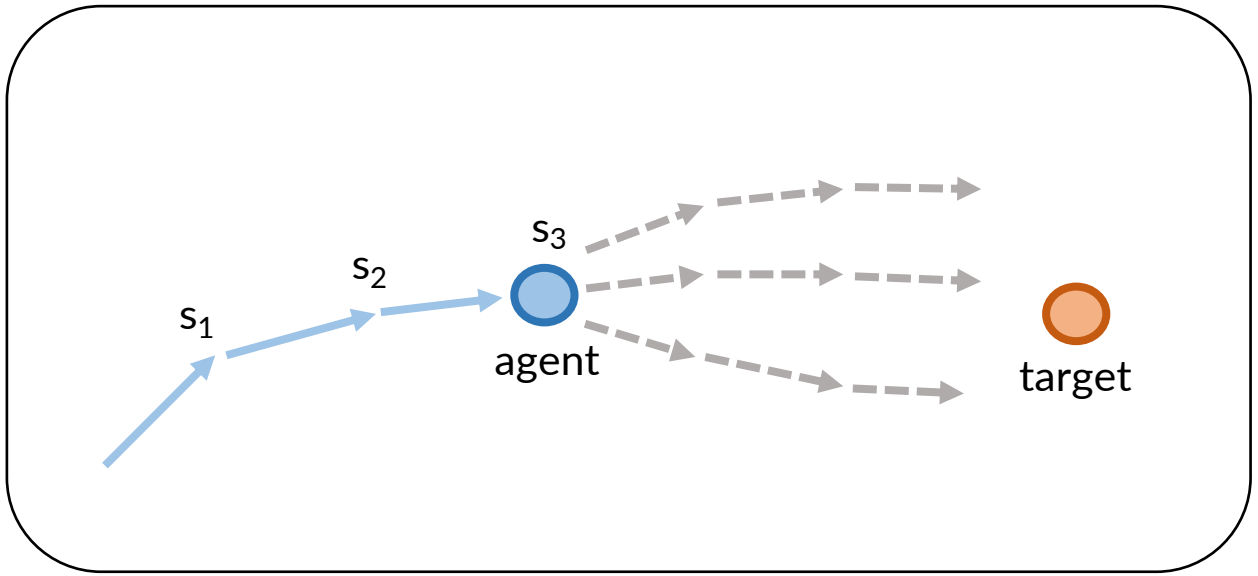


Rajeswaran et al. NeurIPS 2017

Model Predictive Control

Model Predictive Control (MPC) is extremely successful in many control applications including robotics

$$
\begin{aligned}
\underset{u_0, u_1, \cdots, u_H}{\text{maximize}} \quad & \mathbb{E}\left[\sum_{k=0}^{H} \gamma^k r(x_k, u_k)\right] && \text{(Trajectory Cost)} \\
\text{subject to} \quad & x_{k+1} \sim P(\cdot|x_k, u_k) && \text{(Dynamics)} \\
& x_0 = s_t && \text{(Initial State)} \\
\text{Result} \quad & a_t \sim \pi_{MPC}(\cdot|s_t) \equiv u_0 && \text{(Solution)}
\end{aligned}
$$



Instance-specific, local, and fully online approach

Limitations:

- Short horizon bias can lead to highly sub-optimal solutions in some applications
- Longer horizons may be intractable due to computational requirements (real-time constraints) or compounding inaccuracies in the model
- Lack of amortized computation can result in wasteful repeated computation for very similar situations
- Often requires more human insights and heuristics to overcome the above limitations

Dynamic Programming with Function Approximation

Fitted value Iteration *(Bertsekas & Tsitsiklis 1996; Riedmiller 2005; Munos & Szepesvari, 2008)*

$V_\theta(\cdot)$: value network with parameters θ;
$\nu \equiv P(s)$: a distribution over states.
$s' \sim P(\cdot|s, a)$: denotes the next state

Dynamic Programming with Function Approximation

Fitted value Iteration *(Bertsekas & Tsitsiklis 1996; Riedmiller 2005; Munos & Szepesvari, 2008)*

$V_\theta(\cdot)$: value network with parameters θ;
$\nu \equiv P(s)$: a distribution over states.
$s' \sim P(\cdot|s,a)$: denotes the next state

FVI proceeds as follows:

1. In iteration k, construct Bellman targets at $s \sim \nu$:

$$y(s) = \max_a \ \mathbb{E}\left[r(s,a) + \gamma V_{\theta_k}(s')\right]$$

Dynamic Programming with Function Approximation

Fitted value Iteration *(Bertsekas & Tsitsiklis 1996; Riedmiller 2005; Munos & Szepesvari, 2008)*

$V_\theta(\cdot)$: value network with parameters θ;
$\nu \equiv P(s)$: a distribution over states.
$s' \sim P(\cdot|s,a)$: denotes the next state

FVI proceeds as follows:

1. In iteration k, construct Bellman targets at $s \sim \nu$:

$$y(s) = \max_a \ \mathbb{E}\left[r(s,a) + \gamma V_{\theta_k}(s')\right]$$

2. Update value network to approximate targets:

$$\theta_{k+1} \in \arg\min_\theta \ \mathbb{E}_{s\sim\nu}\left[\left(V_\theta(s) - y(s)\right)^2\right]$$

After sufficient iterations, recover policy as:

$$\pi_{FVI}(s) = \arg\max_a \ \mathbb{E}\left[r(s,a) + \gamma V_\theta(s')\right]$$

Dynamic Programming with Function Approximation

Fitted value Iteration *(Bertsekas & Tsitsiklis 1996; Riedmiller 2005; Munos & Szepesvari, 2008)*

$V_\theta(\cdot)$: value network with parameters θ;
$\nu \equiv P(s)$: a distribution over states.
$s' \sim P(\cdot|s,a)$: denotes the next state

FVI proceeds as follows:

1. In iteration k, construct Bellman targets at $s \sim \nu$:

$$y(s) = \max_a \ \mathbb{E}\left[r(s,a) + \gamma V_{\theta_k}(s')\right]$$

2. Update value network to approximate targets:

$$\theta_{k+1} \in \arg\min_\theta \ \mathbb{E}_{s\sim\nu}\left[\left(V_\theta(s) - y(s)\right)^2\right]$$

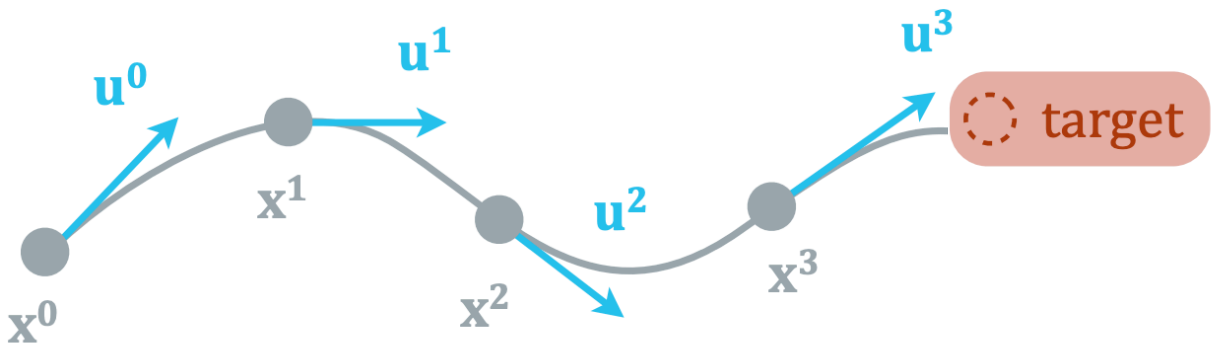
After sufficient iterations, recover policy as:

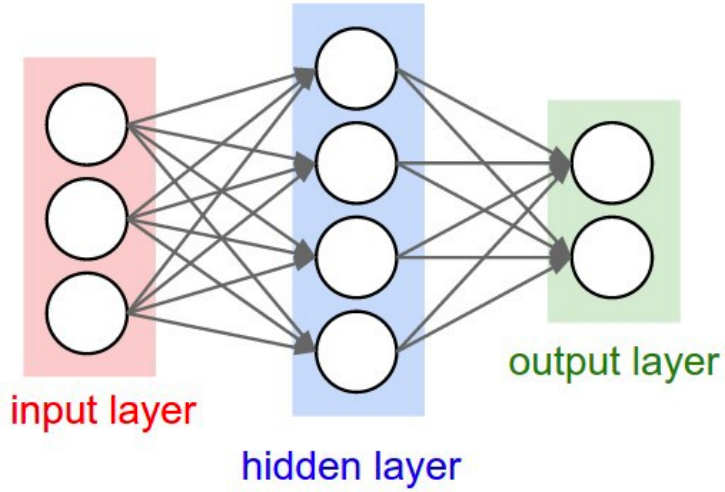
$$\pi_{FVI}(s) = \arg\max_a \ \mathbb{E}\left[r(s,a) + \gamma V_\theta(s')\right]$$

Challenges

1. *Error amplification:* errors in value approximation translate to policy in a *worst-case* way.
2. *Non-stationarity of targets:* Bellman targets change across iterations, as a result may not converge in practice (manifests as chattering)
3. *Choice of measure:* picking a good state distribution critical for propagation of information. Analogous to exploration.

POLO : MPC + Value Function Learning





Model-based trajectory optimization (aka MPC):
instance specific, short-horizon bias,
but efficient optimization and
good out of distribution performance

Learning and function approximation:
global and reusable,
but error amplification, inefficient to train,
and poor out of distribution performance

Can we combine their benefits and remove their limitations?

POLO : MPC + Value Function Learning

Acting in the world: Online MPC with terminal value

$$\begin{aligned} \underset{u_0,u_1,\cdots,u_H}{\text{maximize}} \quad & \mathbb{E}\left[\sum_{k=0}^{H-1} \gamma^k r(x_k,u_k) + \gamma^H V_\theta(x_H)\right] \\ \text{subject to} \quad & x_{k+1} \sim P(\cdot|x_k,u_k) \\ & x_0 = s_t \\ \text{Result} \quad & a_t \sim \pi_{MPC}(\cdot|s_t) \equiv u_0 \end{aligned}$$

- Retains all benefits of online optimization and also gains long horizon foresight through value function
- Value function learning stabilized due to N-step targets
- Error amplification minimized due MPC at run-time

But what about exploration?

Offline learning: Bellman backups using collected transitions (in replay buffer) with N-step targets

$\mathcal{B} := \{(s,a,r,s')\}$: replay buffer of all transitions so far. Update value function and send to MPC every T steps.

For $k = 1, 2, \cdots$ till convergence:

1. Construct Bellman targets at $s \sim \mathcal{B}$:

$$y(s) = \max_{u_0,\cdots,u_N|x_0=s} \mathbb{E}\left[\sum_{k=0}^{N-1} \gamma^k r(x_k,u_k) + \gamma^N V_{\theta_k}(x_N)\right]$$

2. Update value network to approximate targets:

$$\theta_{k+1} \in \arg\min_\theta \ \mathbb{E}_{s\sim\nu}\left[\left(V_\theta(s) - y(s)\right)^2\right]$$

After sufficient iterations, update MPC value function

POLO : Directed Exploration through Planning

Sketch of Exploration Scheme:

1. ***Model uncertainty*** in value estimation using a collection of value networks (as opposed to one)

   Independently initialized models trained on different mini-batch sequences produce predictions from posterior. Formally true in linear case, empirically good in nonlinear case. (Osband et al. 2018).

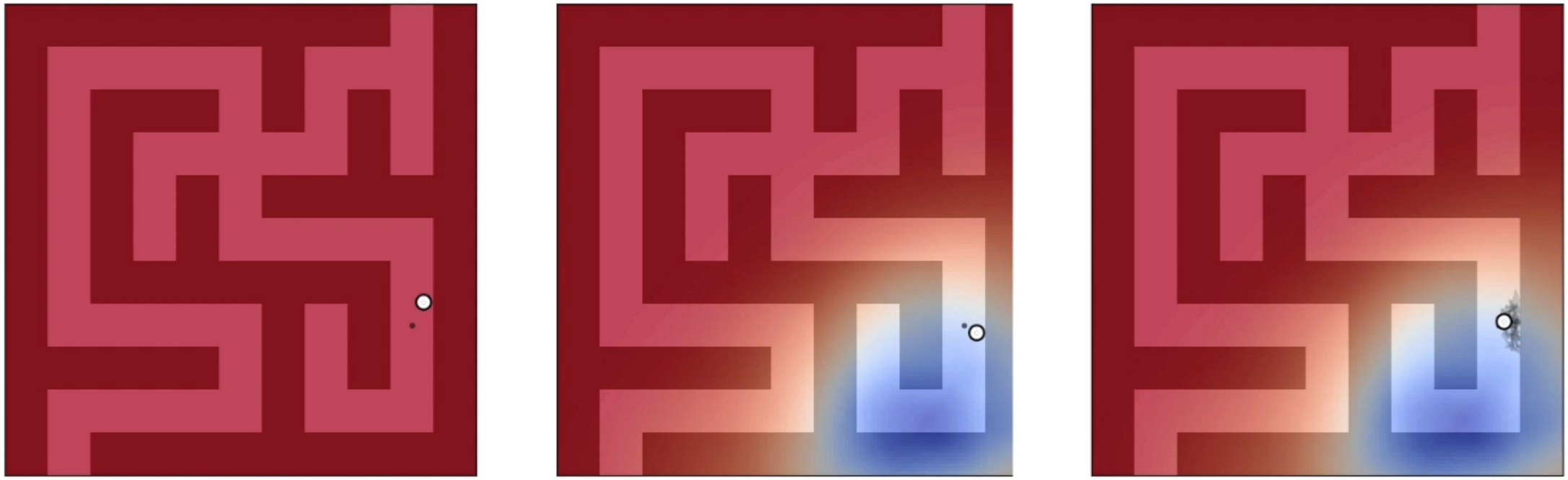
2. ***Optimistic terminal value*** function encourages visiting states with high predicted reward

$$V(s) := \sum_{i=1}^{M} \omega_i(s) V_{\theta_i}(s), \quad \text{where} \quad \omega_i(s) \overset{\text{def}}{:=} \frac{\exp\left(\kappa V_{\theta_i}(s)\right)}{\sum_{j=1}^{M} \exp\left(\kappa V_{\theta_j}(s)\right)}$$

3. ***Temporally coordinated and planned exploration*** emerges due to MPC (optimal hypothesis testing)

Reasoning about uncertainty in value/reward and planning are crucial for efficient exploration

Reward-Free Operation

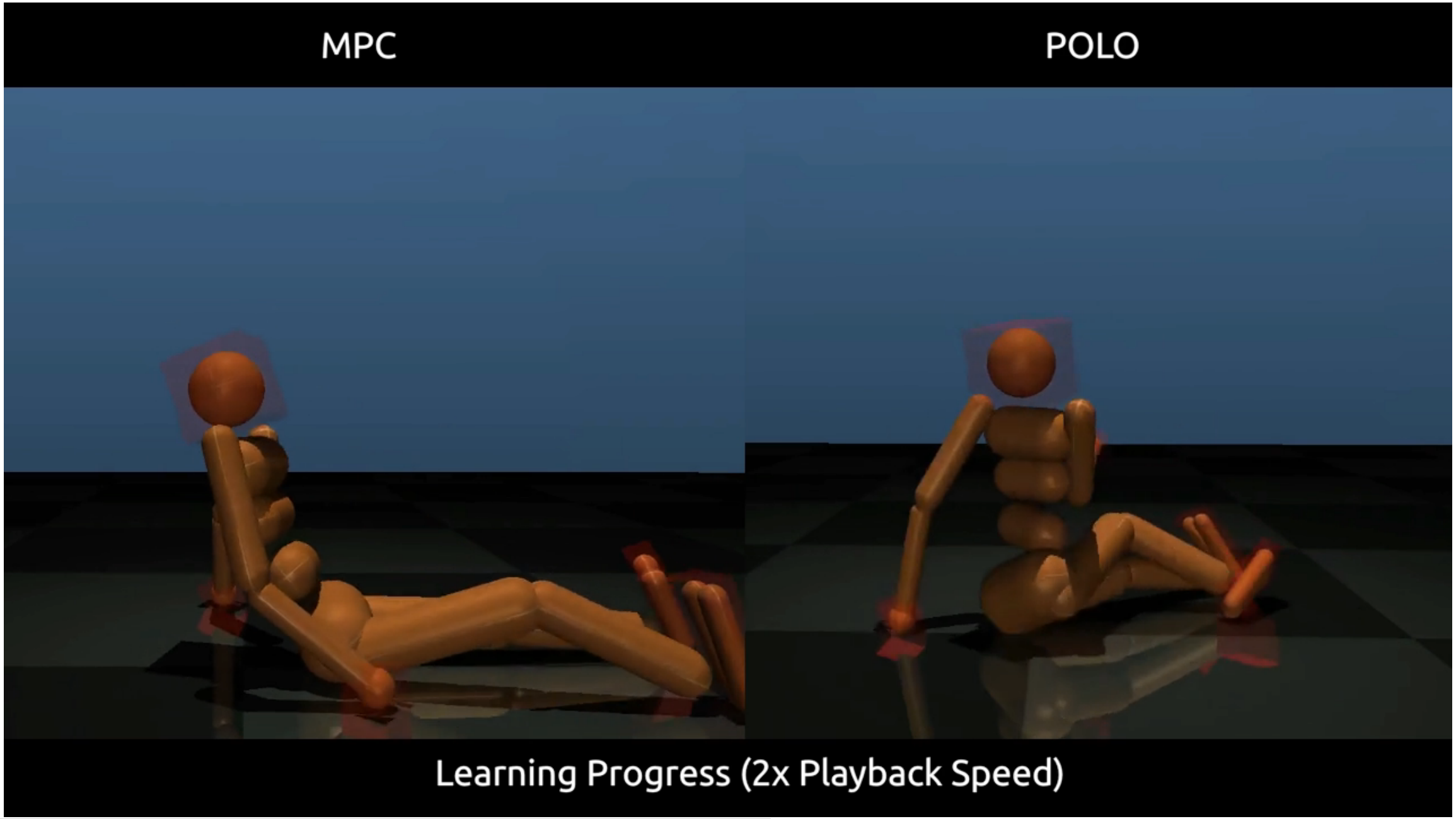


Random

1-Step

32-Step

Lowrey*, Rajeswaran*, Kakade, Todorov, Mordatch. ICLR 2019.



Lowrey*, Rajeswaran*, Kakade, Todorov, Mordatch. ICLR 2019.

Acknowledgements


Kendall Lowrey


Aravind Rajeswaran


Sham Kakade


Emo Todorov


Igor Mordatch

Lowrey*, Rajeswaran*, Kakade, Todorov, Mordatch. Plan Online, Learn Offline:

Efficient Learning and Exploration via Model-Based Control. In ICLR 2019.

Thank you! Questions?